On the Parallelization of Vector Fitting Algorithms

On the Parallelization of Vector Fitting Algorithms

Alessandro Chinea and Stefano Grivet-Talocia, *Senior Member, IEEE*

***Abstract*—The so-called Vector Fitting (VF) algorithm has gained much popularity over the last few years. This technique provides a very effective system identification tool that, starting from input-output responses of a linear and time-invariant system, computes a rational approximation of its transfer matrix. The latter is routinely used to synthesize compact broadband equivalent circuits or state-space models of possibly complex interconnects at the chip, package, board, or even system level. The VF algorithm is based on a combination of iterative linear least squares solutions and eigensolutions, and proves robust and reliable. A potential weak point of VF is its relatively poor scalability with the complexity of the structure under modeling. When the number of input-output ports is very large (one hundred or more, as in the case of power buses or packages), the excessive computational requirements may hinder VF performance and prevent its successful application. In this paper, we address these issues by first presenting a detailed analysis of the computational cost of all the algorithm parts. The results show a very good potential for VF parallelization for multicore hardware, and suggest a few alternative parallelization strategies. Each of these strategies is described in detail. Finally, numerical results and comparisons are provided on a large set of industrial benchmarks. These results demonstrate excellent scalability and speedup factors for the parallel sections of the algorithm, leading to a drastic reduction in overall runtime.**

I. Introduction

This paper discusses about the efficient extraction of behavioral models of linear and time-invariant structures, with particular reference to electrical interconnects such as packages, power buses, connectors, vias and via fields. Despite the huge literature on the subject and the very good results achieved by the academic and industrial communities over the last few years, the model extraction process still remains a hot topic due to the ever increasing demand for higher complexity, higher capacity, higher efficiency and runtime reduction.

Let us consider a generic electrical interconnect with its set of well-defined interface ports. Standard modeling flows involve a preliminary full-wave characterization, aimed at capturing all interactions between port signals, induced fields, geometry, and materials. The results of such characterization can be exploited in two forms. If one has access to the internal variables of the field solver, a standard model order reduction process can be applied [1]-[13], leading to a compact state-space or descriptor model, which can be directly used in time-domain simulation. An alternative is to extract an external port characterization from the field solution, typically as a table of frequency samples of the scattering matrix over the desired bandwidth. This dataset is then fed to a system identification algorithm, which fits a rational model and synthesizes it as a state-space system or as an equivalent circuit [14]-[24], ready for transient analysis. The above techniques are well-established. Yet, there is still margin for improvements.

In this work, we will concentrate on black-box identification, assuming that the system is known via finite frequency samples of its input-output transfer matrix. In particular, we will focus on the Vector Fitting (VF) algorithm. Since its introduction [14], VF has gained much popularity due to its simplicity, robustness and higher performance with respect to the preexisting state of the art methods. The VF scheme computes poles and residues of a rational function providing a fit to the raw data points. The direct nonlinear optimization process is relaxed to a weighted linear least squares system, which is reformulated and solved iteratively until convergence. As opposed to more traditional approaches, VF allows for high-order model identification due to the good conditioning of its "basis" functions, consisting of suitably defined partial fractions. For the above reasons, most commercially available black-box model extraction tools include some implementation of the VF algorithm. We remark that other competitive solutions for black-box identification exist, such as the Löwner matrix approach of [25]. Such methods are not addressed in this work.

A potential weak point of the basic VF scheme is its relatively poor scalability with the number of input-output ports P of the structure under modeling. Although the numerical performance will depend on the particular formulation and implementation of VF, overall complexity will be obviously not less than $O(P^2)$ in the general case. This scaling law may be a serious limit when the number of ports grows beyond one hundred or more, as required by most challenging high-speed packaging applications. In this work, we aim at breaking this complexity by reformulating the VF scheme in such a way that it can be easily parallelized and deployed on multicore hardware platforms. Multicore parallelization is indeed a most promising solution for runtime reduction of numerical simulation or extraction algorithms, since parallel hardware is becoming widespread even at the desktop level and at quite acceptable costs.

After a review of the basic VF scheme in Section II in order to set the notation, we perform in Section III a detailed analysis of the computational cost for each sub-task involved in the VF flow. In particular, we analyze the number of floating point operations (flops) required in case the rational model is structured with common poles among all responses, private poles for each response, and in the intermediate case of common poles for partial subsets of port responses. These three "splitting" strategies have an influence not only on the synthesized model structure, hence on its efficiency in subsequent system-level simulations, but also on the flops required by its identification. The results of this analysis provide the

Manuscript received ; revised.

Alessandro Chinea and Stefano Grivet-Talocia are with the Department of Electronics, Politecnico di Torino, Torino 10129, Italy (e-mail: alessandro.chinea@polito.it, stefano.grivet@polito.it).

basis for the design of suitable parallelization strategies. In particular, we observe that a multiple QR factorization stage is the most demanding part of the scheme, therefore the main candidate for our parallelization. Combining this with the above splitting strategies, we obtain four different parallel versions of VF, which are described in detail in Section IV. A preliminary formulation of one of these versions is available in [26].

The theoretical performance of the presented Parallel VF (PVF) schemes is compared in Section V to the numerical results obtained on a large set of industrial benchmarks (various types of electrical interconnects such as packages, connectors, vias and via fields, power buses). Main parameters controlling the complexity of such test cases is the number of ports P (ranging from 22 to 245), the number of poles per response N (4 to 64), and the number of raw frequency samples K (10 to 1228). These benchmarks, summarized in Table I, form a quite comprehensive and representative set. We remark that N denotes the number of poles that are required for an accurate rational approximation of each scalar transfer matrix element. The actual dynamic order of the complete macromodel will depend on the size of an associated minimal state-space realization, as discussed in Section II-D.

The PVF schemes have been coded for a shared memory architecture using OpenMP and deployed on a 16-core machine. The scalability tests described in Section V show that the parallel efficiency and the speedup factor with respect to a serial implementation are close to ideal for the code fragments that have been parallelized. Obviously, some overhead remains for the small parts of the VF algorithm that cannot be parallelized, due to its iterative nature. Nonetheless, this paper demonstrates that VF parallelization is indeed quite effective, leading to major runtime reduction and in some cases to almost real-time model extraction.

Before proceeding, we remark that no discussion will be performed here on the important subject of model passivity check and enforcement [27], [28], [29]. Several methods are available for this task [30]-[44]. Some preliminary results on their parallelization are available in [45]. A comprehensive treatment of parallelized passivity check and enforcement schemes will be the subject of a future report.

TABLE I
LIST OF BENCHMARK CASES. P DENOTES THE NUMBER OF PORTS, K THE NUMBER OF FREQUENCY SAMPLES, AND N THE NUMBER OF POLES PER RESPONSE.

Id	Ports P	Samples K	Poles N
1	56	1001	50
2	83	1228	30
3	245	197	20
4	34	570	64
5	48	690	26
6	92	71	22
7	40	1001	10
8	46	71	28
9	34	1000	10
10	167	27	12
11	41	572	10
12	160	101	6
13	44	301	12
14	24	1001	12
15	35	145	20
16	22	201	22
17	22	201	20
18	28	501	6
19	25	572	6
20	23	572	6
21	40	40	12
22	140	10	6
23	72	11	4
24	52	13	4

II. BACKGROUND

A. The basic VF scheme

The Vector Fitting (VF) scheme [14] is an efficient algorithm for the estimation of the rational transfer function of a linear multiport structure. We start by reviewing the basic VF scheme in order to set the notation. In particular, we adopt a formulation among the many available in the literature [14]-[24], that will facilitate the description of our parallelization work. The various algorithm steps that are outlined below are also summarized in the block-diagram of Fig. 2.

The input to the VF algorithm is a set of simulated or measured frequency samples $S_{ij,k} = S_{ij}(\mathrm{j}\omega_k)$, with $k = 1, 2, \ldots, K$ and $i, j = 1, 2, \ldots, P$, where P denotes the number of electrical ports. Assuming a macromodel in rational form and cast as a sum of partial fractions,

$$\tilde{S}_{ij}(s) = \tilde{r}_{ij,0} + \sum_{n=1}^{N} \frac{\tilde{r}_{ij,n}}{s - \tilde{p}_n}, \tag{1}$$

the VF algorithm tries to find the set of unknown poles $\tilde{p}_n$ and residues $\tilde{r}_{ij,n}$ that minimize the least squares distances $\sum_{k=1}^{K} |S_{ij,k} - \tilde{S}_{ij}(\mathrm{j}\omega_k)|^2$.

The non linear optimization problem (1) is solved by an iterative procedure that refines an initial estimate or guess p_n of the poles (the so-called "starting poles"). This relaxation is achieved by introducing a weight function

$$\sigma(s) = 1 + \sum_{n=1}^{N} \frac{c_n}{s - p_n} = \frac{\prod_{n=1}^{N}(s - z_n)}{\prod_{n=1}^{N}(s - p_n)} \tag{2}$$

with known poles p_n and unknown residues c_n. The following condition

$$\sigma(\mathrm{j}\omega_k) S_{ij,k} \simeq r_{ij,0} + \sum_{n=1}^{N} \frac{r_{ij,n}}{\mathrm{j}\omega_k - p_n}. \tag{3}$$

is then enforced by collecting all responses $i, j \in \{1, 2, \ldots, P\}$ and all frequency samples $k \in \{1, 2, \ldots, K\}$ in a single linear system, which is solved in least squares sense.

The solution of (3) gives the residues c_n of the weight function and the coefficients $r_{ij,n}$, which are discarded. Intuitively, if the raw data samples $S_{ij,k}$ come from a N-th order rational function and if (3) is solved exactly, the zeros z_n of the weight function must implicitly cancel the (still unknown) true poles of the model. This observation provides the guideline to setup the next VF iteration, i.e., to replace the starting poles of $\sigma(s)$ with its zeros $p_n \leftarrow z_n$, and to repeat the process until convergence. The zeros z_n can be found by computing the eigenvalues

$$z_n = \mathrm{eig}\left\{ \boldsymbol{A}_\sigma - \boldsymbol{B}_\sigma \boldsymbol{D}_\sigma^{-1} \boldsymbol{C}_\sigma \right\}, \tag{4}$$

where $(\boldsymbol{A}_\sigma, \boldsymbol{B}_\sigma, \boldsymbol{C}_\sigma, \boldsymbol{D}_\sigma)$ is a minimal state-space realization of $\sigma(s)$. Although a general convergence proof is not yet available [46], the p_n usually converge to the dominant poles of the structure $\tilde{p}_n$ in few iterations.

Once the dominant poles $\tilde{p}_n$ have been found, another simple linear least squares problem $\forall i, j \in \{1, 2, \dots, P\}, k \in \{1, 2, \dots, K\}$

$$S_{ij,k} \simeq \tilde{r}_{ij,0} + \sum_{n=1}^{N} \frac{\tilde{r}_{ij,n}}{\mathrm{j}\omega_k - \tilde{p}_n}, \tag{5}$$

is used to determine the residues $\tilde{r}_{ij,n}$.

B. The pole identification stage

A closer look at the problem (3) reveals the matrix structure

$$\begin{bmatrix} \boldsymbol{\varphi}_1 & \mathbf{0} & \dots & \mathbf{0} & -\boldsymbol{S}_1\boldsymbol{\varphi} \\ \mathbf{0} & \boldsymbol{\varphi}_1 & \dots & \mathbf{0} & -\boldsymbol{S}_2\boldsymbol{\varphi} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\varphi}_1 & -\boldsymbol{S}_{P^2}\boldsymbol{\varphi} \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_{P^2} \\ \boldsymbol{c} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_1\mathbf{1}_K \\ \boldsymbol{S}_2\mathbf{1}_K \\ \vdots \\ \boldsymbol{S}_{P^2}\mathbf{1}_K \end{bmatrix} \tag{6}$$

where $\mathbf{1}_K \in \mathbb{R}^K$ is a column vector of ones, and

$$\begin{aligned} \boldsymbol{\varphi} &= \begin{bmatrix} \frac{1}{\mathrm{j}\omega_1 - p_1} & \frac{1}{\mathrm{j}\omega_1 - p_2} & \cdots & \frac{1}{\mathrm{j}\omega_1 - p_N} \\ \frac{1}{\mathrm{j}\omega_2 - p_1} & \frac{1}{\mathrm{j}\omega_2 - p_2} & \cdots & \frac{1}{\mathrm{j}\omega_2 - p_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\mathrm{j}\omega_K - p_1} & \frac{1}{\mathrm{j}\omega_K - p_2} & \cdots & \frac{1}{\mathrm{j}\omega_K - p_N} \end{bmatrix}, \\ \boldsymbol{\varphi}_1 &= \begin{bmatrix} \mathbf{1}_K & \boldsymbol{\varphi} \end{bmatrix} \\ \boldsymbol{S}_{i+(P-1)j} &= \operatorname*{diag}_{k \in \{1,2,\dots,K\}} (S_{ij,k}) \\ \boldsymbol{r}_{i+(P-1)j} &= \begin{bmatrix} r_{ij,0} & r_{ij,1} & \cdots & r_{ij,N} \end{bmatrix}^T \\ \boldsymbol{c} &= \begin{bmatrix} c_1 & c_2 & \cdots & c_N \end{bmatrix}^T. \end{aligned} \tag{7}$$

The matrix involved in (6) is depicted in Fig. 1a and has a size $KP^2 \times ((N+1)P^2 + N)$. This size is obviously prohibitively large for structures with few tens of ports. For example, case #5 of Table I ($P = 48$, $K = 690$, $N = 26$) would require about 1.4 TB of memory (using double precision complex numbers). By using a sparse storage format, the required memory decreases to 1.3 GB, but even in this case a direct solution of (6) requires an excessive computational cost.

In [24], a new formulation of the least squares problem (6) is introduced in order to overcome these limitations. Given that the terms $\boldsymbol{r}_l$ are discarded in the VF process, the main idea in [24] is to find a new set of equations where only the unknowns $\boldsymbol{c}$ are considered and solved for. First, each block-row of system (6) for $l = 1, 2, \dots, P^2$ is considered independently (see Fig. 1b) and subject to a "thin" QR decomposition [47] (Fig. 1c)

$$\begin{bmatrix} \boldsymbol{\varphi}_1 & -\boldsymbol{S}_l\boldsymbol{\varphi} \end{bmatrix} = \boldsymbol{\mathcal{Q}}_l\boldsymbol{\mathcal{R}}_l = \boldsymbol{\mathcal{Q}}_l \begin{bmatrix} \boldsymbol{\mathcal{R}}_l^{11} & \boldsymbol{\mathcal{R}}_l^{12} \\ \mathbf{0} & \boldsymbol{\mathcal{R}}_l^{22} \end{bmatrix}, \tag{8}$$

where the $N \times N$ upper triangular matrix $\boldsymbol{R}_l^{22}$ is associated to the unknowns $\boldsymbol{c}$. By collecting all the matrices $\boldsymbol{R}_l^{22}$ we obtain

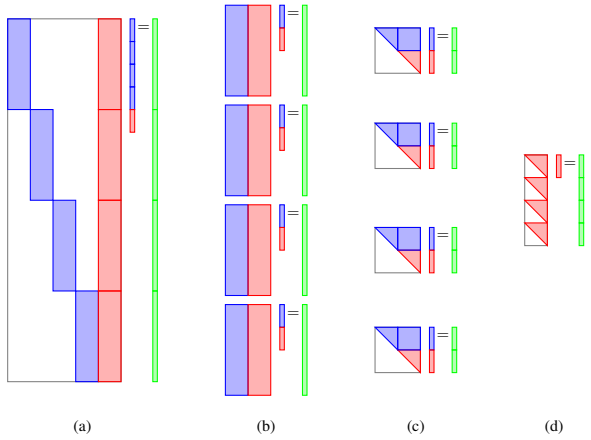

Fig. 1. (a) Sketch of the matrix structure for the pole identification system (6); (b) problem decoupling and (c) associated QR decompositions; (d) compressed system (9).

the new "compressed" least squares problem

$$\begin{bmatrix} \boldsymbol{\mathcal{R}}_1^{22} \\ \boldsymbol{\mathcal{R}}_2^{22} \\ \vdots \\ \boldsymbol{\mathcal{R}}_{P^2}^{22} \end{bmatrix} \boldsymbol{c} = \begin{bmatrix} \boldsymbol{\mathcal{Q}}_1^T\boldsymbol{S}_1\mathbf{1}_K \\ \boldsymbol{\mathcal{Q}}_2^T\boldsymbol{S}_2\mathbf{1}_K \\ \vdots \\ \boldsymbol{\mathcal{Q}}_{P^2}^T\boldsymbol{S}_{P^2}\mathbf{1}_K \end{bmatrix}, \tag{9}$$

depicted in Fig. 1d, that requires much less memory ($NP^2 \times N$) and is computationally tractable.

C. Relaxed Vector Fitting

We recall in this section a slightly modified version of VF which has superior convergence performances, especially in presence of noise in the raw data [17]. The main idea is to remove the asymptotic unitary constraint of $\sigma(s)$ in (2) by replacing it with a decision variable d, so we define a modified weight function

$$\widehat{\sigma}(s) = d + \sum_{n=1}^{N} \frac{c_n}{s - p_n} = d \frac{\prod_{n=1}^{N}(s - z_n)}{\prod_{n=1}^{N}(s - p_n)} \tag{10}$$

In this case the least squares system in (3) becomes

$$\begin{bmatrix} \boldsymbol{\varphi}_1 & \mathbf{0} & \dots & \mathbf{0} & -\boldsymbol{S}_1\boldsymbol{\varphi}_1 \\ \mathbf{0} & \boldsymbol{\varphi}_1 & \dots & \mathbf{0} & -\boldsymbol{S}_2\boldsymbol{\varphi}_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\varphi}_1 & -\boldsymbol{S}_{P^2}\boldsymbol{\varphi}_1 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_{P^2} \\ \hat{\boldsymbol{c}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \tag{11}$$

with

$$\hat{\boldsymbol{c}} = \begin{bmatrix} d & c_1 & c_2 & \cdots & c_N \end{bmatrix}^T. \tag{12}$$

In order to avoid the all-vanishing trivial solution one more equation is added,

$$\operatorname{Re}\left\{ \sum_{k=1}^{K} \left(d + \sum_{n=1}^{N} \frac{c_n}{\mathrm{j}\omega_k - p_n} \right) \right\} = K, \tag{13}$$

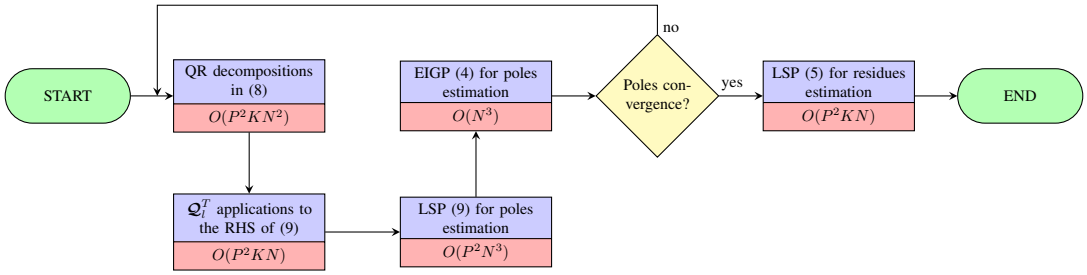

Fig. 2. Block-diagram description of the adopted formulation of the VF algorithm, including the computational cost of each block. Acronyms LSP and EIGP stand for Least Squares Problem and Eigenvalue Problem, respectively. Note that the block with the applications of $\boldsymbol{\mathcal{Q}}_l^T$ is not necessary in the relaxed version of the VF algorithm.

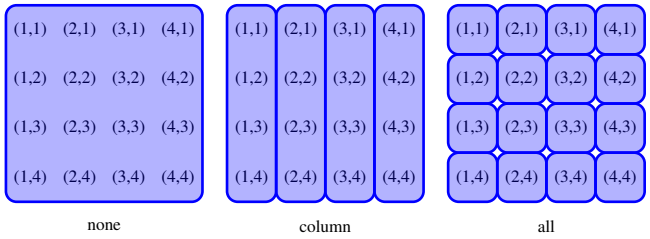

Fig. 3. Graphical illustration of the adopted splitting strategies. The "none" case considers all responses together; the "column" case splits the system into a set of disjoint single-input subsystems corresponding to the columns of the transfer matrix; the "all" case models each response independently.

According to this relaxed formulation, the QR decompositions are applied to

$$\begin{bmatrix}\boldsymbol{\varphi}_1 & -\boldsymbol{S}_l\boldsymbol{\varphi}_1\end{bmatrix} = \widehat{\boldsymbol{\mathcal{Q}}}_l\widehat{\boldsymbol{\mathcal{R}}}_l = \widehat{\boldsymbol{\mathcal{Q}}}_l\begin{bmatrix}\widehat{\boldsymbol{\mathcal{R}}}_l^{11} & \widehat{\boldsymbol{\mathcal{R}}}_l^{12} \\ \mathbf{0} & \widehat{\boldsymbol{\mathcal{R}}}_l^{22}\end{bmatrix}, \tag{14}$$

and the final system for poles identification becomes

$$\begin{bmatrix}\boldsymbol{\mathcal{R}}_1^{22} \\ \boldsymbol{\mathcal{R}}_2^{22} \\ \vdots \\ \boldsymbol{\mathcal{R}}_{P^2}^{22} \\ \boldsymbol{r}^T\end{bmatrix}\hat{\boldsymbol{c}} = \begin{bmatrix}\mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ K\end{bmatrix} \tag{15}$$

where the row vector $\boldsymbol{r}^T$ contains the coefficients of the constraint in (13).

D. Splitting strategies

The derivation of the basic VF scheme in Section II-A assumes that the poles p_n are common to all model responses, see (1). This assumption may be appropriate when modeling lumped circuits or structures with global resonances. When however VF is used as a general numeric tool for rational approximation, the common-pole assumption may be overly restrictive and may lead to non-optimal accuracy. Another drawback in the common-pole assumption is the necessity to consider all responses together in the pole identification stage, as discussed in Section II-B, leading to large-size systems whose solution requires significant CPU and memory resources.

A different approach is to consider a large multiport structure as a collection of connected but disjoint subsystems. For instance, considering the columns of the transfer matrix separately corresponds to a partition of the system into a set of Single-Input-Multiple-Output (SIMO) subsystems. The connection of these subsystems into a global model leads to state-space realizations with higher degree of sparsity [15], [31], with increased performance in successive transient simulations. For what concerns the model identification stage, each of these subsystems can have its own private set of poles, which can thus be computed using the VF scheme on a restricted set of responses. This procedure allows saving memory and CPU usage. Moreover, such a system partition is an obvious and effective decoupling strategy in view of a parallelization of the overall model extraction process.

We provide a more formal description of this splitting strategy. Let $\mathcal{I}$ be the set of P^2 pairs (i,j), with $i,j \in \{1,\dots P\}$, and let $\mathcal{J} = \{\mathcal{I}_1,\dots,\mathcal{I}_M\}$ be a partition of $\mathcal{I}$, i.e.,

$$\begin{aligned}&\bigcup_{\mu=1}^{M}\mathcal{I}_\mu = \mathcal{I}\,,\\ &\mathcal{I}_\mu \cap \mathcal{I}_\nu = \emptyset, \quad \forall \mu,\nu \in \{1,\dots M\}, \quad \mu \neq \nu\,.\end{aligned} \tag{16}$$

Each element $\mathcal{I}_\mu$ in this partition is associated with a submodel, whose responses share a common pole set. This submodel reads

$$\tilde{S}_{ij}(s) = \tilde{r}_{ij,0} + \sum_{n=1}^{N_\mu}\frac{\tilde{r}_{ij,n}}{s-\tilde{p}_{\mu,n}}, \quad \forall (i,j) \in \mathcal{I}_\mu\,. \tag{17}$$

The identification of this μ-th submodel is performed as described in Section II-B, but discarding the responses $(i,j) \notin \mathcal{I}_\mu$ in all steps.

Although quite general and fancy splitting strategies can be devised, in this work we use only the following three alternatives, which we consider practically relevant. See also Fig. 3 for a graphical illustration.

- **none**: a single partition $\mathcal{J} = \{\mathcal{I}\}$ is used, i.e. all responses are considered as a whole;
- **column**: the transfer matrix is partitioned column-wise, i.e., $\mathcal{J} = \{\mathcal{I}_1,\dots,\mathcal{I}_P\}$ with $\mathcal{I}_\mu = \{(1,\mu),\dots,(P,\mu)\}$;
- **all**: each partition is a singleton, i.e., $\mathcal{J} = \{\mathcal{I}_1,\dots,\mathcal{I}_{P^2}\}$, with $\mathcal{I}_{i+(P-1)j} = \{(i,j)\}$.

Usage of the keywords that label each partitioning scheme will be consistent throughout the paper.

A few remarks on the state-space realization $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ of models obtained using the above splitting strategies are in order. In the "all" case, the state-space dimension is obviously $\sum_{\mu=1}^{P^2} N_\mu$, which reduces to $P^2 N$ if all responses are fitted with the same number $N_\mu = N$ of poles. This large state-space dimension is compensated by the high sparsity of the state-space matrices $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$, which are block-diagonal. For the "column" case, matrix $\boldsymbol{A}$ is block-diagonal with size $\sum_{\mu=1}^{P} N_\mu$, matrix $\boldsymbol{B}$ is a sparse input-to-state mapping and matrix $\boldsymbol{C}$ stores all residues and is full [31]. Finally, for the "none" case, minimal realizations can be obtained using the Gilbert form, following the procedure reported in [48], [49]. In case of full-rank residues, the same state-space size of the "column" case is obtained. Otherwise, the state-space size results $\sum_{n=1}^{N} \rho_n$, with $\rho_n \leq P$ denoting the rank of the residue matrix associated to the n-th pole.

The actual numerical efficiency in system-level simulation of models in the three above formats strongly depends on the implementation. For instance, a direct equivalent circuit synthesis for SPICE-based simulation results more compact for the "none" and "column" cases, due to the reduced number of required dynamic components. However, for more advanced and direct implementations making use of analytic Laplace transform inversion and fast/recursive convolutions (an example is the Laplace element [50], now available in most commercial circuit solvers), all model forms are essentially equivalent.

III. Complexity analysis

A detailed analysis of the computational cost required by the various VF steps is the key to devise a proper parallelization strategy. We recall first the number of floating-point operations (flops) needed for some standard linear algebra problems involved in VF algorithm [47]. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, the QR factorization of $\boldsymbol{A}$ is performed in

$$\mathrm{Fl}_{\mathrm{QR}}(m,n) = \begin{cases} 2m^2 n - \dfrac{2}{3}m^3 & \text{if } n \geqslant m \\ 2n^2 m - \dfrac{2}{3}n^3 & \text{if } n < m \end{cases} \tag{18}$$

flops. An accurate description of flops required for a standard eigenvalue problem is more difficult to find, given that this depends on how rapidly the algorithm converges [47]. An average number is

$$\mathrm{Fl}_{\mathrm{EIG}}(m) \sim 10m^3 \tag{19}$$

where m is the size of the eigenvalue problem.

We consider now a least squares problem with multiple right-hand sides

$$\begin{aligned} &\min \|\boldsymbol{A}\boldsymbol{x}_1 - \boldsymbol{b}_1\| \\ &\min \|\boldsymbol{A}\boldsymbol{x}_2 - \boldsymbol{b}_2\| \\ &\quad\vdots \\ &\min \|\boldsymbol{A}\boldsymbol{x}_p - \boldsymbol{b}_p\| \end{aligned} \tag{20}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ (with $m > n$), $\boldsymbol{b}_k \in \mathbb{R}^m$, $\boldsymbol{x} \in \mathbb{R}^n$. The solution of (20) is performed in three steps. First, a QR decomposition of $\boldsymbol{A}$ is computed

$$\boldsymbol{A} = \boldsymbol{\mathcal{Q}}\boldsymbol{\mathcal{R}}, \tag{21}$$

then each right-hand side is multiplied by $\boldsymbol{\mathcal{Q}}^T$,

$$\hat{\boldsymbol{b}}_k = \boldsymbol{\mathcal{Q}}^T \boldsymbol{b}_k \tag{22}$$

and finally the triangular systems

$$\boldsymbol{\mathcal{R}}\boldsymbol{x}_k = \hat{\boldsymbol{b}}_k \tag{23}$$

are solved using back-substitution. The three steps require $\mathrm{Fl}_{\mathrm{QR}}(m,n)$, $2mnp$ and $n^2 p$ flops, respectively. Therefore, the total number of floating point operations for a least squares problem with multiple right hand sides is

$$\mathrm{Fl}_{\mathrm{LSP}}(m,n,p) = \mathrm{Fl}_{\mathrm{QR}}(m,n) + p(2mn + n^2) \tag{24}$$

With these premises, we are ready to count the number of floating point operations required by the VF algorithm, a block diagram of which is reported for clarity in Fig. 2. We indicate with H the number of iterations required for poles estimation and with L the number of elements of the transfer matrix to be modeled. We suppose also that the relaxed version of VF is used, therefore the block of $\boldsymbol{\mathcal{Q}}_l^T$ applications in Figure 2 is not necessary. The number of flops results

$$\begin{aligned} \mathrm{Fl}_{\mathrm{VF}}(L,K,N,H) = H\big[&L \cdot \mathrm{Fl}_{\mathrm{QR}}(2K, 2(N+1)) \\ &+ \mathrm{Fl}_{\mathrm{LSP}}((N+1)L+1, N+1, 1) \\ &+ \mathrm{Fl}_{\mathrm{EIG}}(N)\big] \\ &+ \mathrm{Fl}_{\mathrm{LSP}}(2K, N+1, L) \end{aligned} \tag{25}$$

where the number of rows in QR decompositions as well as the number of rows in final LS problem are multiplied by two because the complex matrices involved in these operations are first converted to double size real matrices by taking their real and imaginary parts.

We now substitute (18), (19), and (24) into (25). Extracting the leading term under the assumption $K > N$ (i.e., the number of poles of the rational approximation does not exceed the number of available frequency samples) leads to the following dominant scaling law

$$\mathrm{Fl}_{\mathrm{VF}}(L,K,N,H) \simeq 16LKN^2H\,. \tag{26}$$

It can be observed that this expression is basically the same as would be obtained by considering the QR factorization alone, which is therefore the dominant part of the VF scheme for what concerns the CPU requirements. Table II confirms this proposition by listing the percentage share of overall CPU time needed by each VF sub-task, based on the above theoretical derivation, and for the "none" splitting strategy. The QR phase (14) needs 87% of the CPU time in the worst-case #22 and requires more than 98% for most cases. The solution of the LSP problem for poles identification (15) requires only 1-2% in most cases, reaching 11% for case #22. The computational cost EIG problem (4) is negligible with respect to all other parts, whereas the final LSP problem (5) for the evaluation of the residues is also in the range 0-2%. Collectively, these results show that the primary goal in our parallelization work will be to address the QR step.

We now consider the influence of the adopted splitting strategy on the theoretical flops count. Considering the general

TABLE II
THEORETICAL PERCENTAGE OF CPU-TIME SPENT ON EACH VF STEP ("NONE" SPLITTING CASE) FOR THE BENCHMARK CASES OF TABLE I.

Id	QR (14)	LSP (15)	EIGP (4)	LSP (5)
1	99.17	0.66	0.00	0.17
2	99.40	0.33	0.00	0.27
3	98.16	1.42	0.00	0.41
4	98.37	1.48	0.01	0.14
5	99.17	0.51	0.00	0.32
6	95.08	4.50	0.00	0.42
7	99.09	0.15	0.00	0.76
8	93.90	5.74	0.01	0.35
9	99.09	0.15	0.00	0.76
10	92.10	7.11	0.00	0.79
11	98.98	0.26	0.00	0.76
12	97.80	0.99	0.00	1.21
13	98.76	0.58	0.00	0.65
14	99.17	0.17	0.00	0.65
15	97.62	1.94	0.01	0.43
16	98.07	1.52	0.01	0.40
17	98.17	1.39	0.01	0.43
18	98.61	0.20	0.00	1.19
19	98.63	0.17	0.00	1.20
20	98.63	0.17	0.00	1.20
21	94.61	4.64	0.01	0.74
22	87.05	11.35	0.00	1.59
23	90.72	7.29	0.00	1.99
24	91.98	6.09	0.00	1.93

case (25), we can particularize it as follows

$$\begin{aligned} \mathrm{Fl}_{\text{VF-NONE}}(P,K,N,H) &= \mathrm{Fl}_{\text{VF}}(P^2,K,N,H)\,, \\ \mathrm{Fl}_{\text{VF-COLUMN}}(P,K,N,H) &= P\cdot\mathrm{Fl}_{\text{VF}}(P,K,N,H)\,, \\ \mathrm{Fl}_{\text{VF-ALL}}(P,K,N,H) &= P^2\cdot\mathrm{Fl}_{\text{VF}}(1,K,N,H)\,. \end{aligned} \tag{27}$$

Due to the linear scaling with respect to the number of responses L, see (26), the difference in performance for a serial implementation of the three splitting strategies is expected to be minimal. Of course, the number of operations required by any splitting strategy will be slightly larger with respect to the case of no splitting, due to the additional overhead introduced by the repeated EIG and LSP problems, which are solved only once per iteration in case of no splitting. Taking $\mathrm{Fl}_{\text{VF-NONE}}(P,K,N,H)$ as a reference and computing the theoretical flops count for the "column" and "all" splitting cases, we obtain a maximum overhead of 1% and 39%, respectively, among all cases of Table I. So, the only motivation for introducing these splitting strategies appears to be the promise for an embarrassingly parallel implementation.

IV. PARALLELIZING VECTOR FITTING

In this work, we focus on a shared-memory programming model, based on the OpenMP paradigm [51]. The main idea is to organize and parallelize the computational tasks by splitting a master thread into a specified number T of slave threads that run concurrently. Each of these threads has an identification number $t \in \{0,1,\ldots,T-1\}$ that is easily retrieved at runtime using the OpenMP programming interface, and which can be used to schedule and assign individual tasks. Parallel blocks in the code are introduced by using suitable preprocessor directives, which specify all the properties of the parallel sections (e.g., shared/private variables, type of scheduling method, etc.). OpenMP has the advantage of requiring very few modifications of the original serial code but, unlike other methods of parallel programming based on message passing (MPI, see [52]), it can be used only in shared memory computers. Two different parallelization strategies of VF are now analyzed.

Algorithm 1 Poles refinement

Input

T : number of available threads
L : number of elements to be processed
N : number of poles
K : number of frequency samples
$\widehat{\mathcal{I}}$: split to be processed, $\widehat{\mathcal{I}} = \{(i_1,j_1),\ldots,(i_L,j_L)\}$
p_n : current poles, $\forall n \in \{1,\ldots,N\}$
ω_k : frequency points, $\forall k \in \{1,\ldots,K\}$
$S_{ij,k}$: frequency samples, $\forall (i,j) \in \widehat{\mathcal{I}}$, $\forall k \in \{1,\ldots,K\}$

function $z_n = \text{POLESREF}(T, L, N, K, \widehat{\mathcal{I}}, p_n, \omega_k, S_{ij,k})$

build matrix $\boldsymbol{\varphi}_1$ using ω_k and p_n
$\tilde{N} \leftarrow N+1$
parallel region (only if $T > 1$)
$t \leftarrow$ thread number
for $l \leftarrow \left\lfloor \frac{t}{T}L \right\rfloor + 1$ **to** $\left\lfloor \frac{t+1}{T}L \right\rfloor$
$\boldsymbol{V}^{(t)}_{[:,1:\tilde{N}]} \leftarrow \boldsymbol{\varphi}_1$
$\boldsymbol{V}^{(t)}_{[:,\tilde{N}+1:2\tilde{N}]} \leftarrow -\operatorname{diag}_{k\in\{1,\ldots,K\}}\{S_{i_l j_l,k}\}\boldsymbol{\varphi}_1$
$\left(\boldsymbol{\mathcal{Q}}^{(t)}, \boldsymbol{\mathcal{R}}^{(t)}\right) \leftarrow \text{QR}\left(\boldsymbol{V}^{(t)}\right)$ (dgeqrf)
$\boldsymbol{W}_{[(l-1)\tilde{N}+1:l\tilde{N},:]} \leftarrow \boldsymbol{\mathcal{R}}^{(t)}_{[\tilde{N}+1:2\tilde{N},\tilde{N}+1:2\tilde{N}]}$
end for
end parallel region
$\boldsymbol{W}_{[L\tilde{N}+1,:]} \leftarrow \boldsymbol{r}^T$
$\boldsymbol{w} \leftarrow \begin{bmatrix} 0 & \cdots & 0 & K \end{bmatrix}^T$
$\hat{\boldsymbol{c}} \leftarrow \arg\min \|\boldsymbol{W}\boldsymbol{x} - \boldsymbol{w}\|$ (dgels)
Build s.s. realization $\{\boldsymbol{A}_\sigma, \boldsymbol{B}_\sigma, \boldsymbol{C}_\sigma, \boldsymbol{D}_\sigma\}$ using $\hat{\boldsymbol{c}}$ and p_n
$z_n \leftarrow \text{eig}\left\{\boldsymbol{A}_\sigma - \boldsymbol{B}_\sigma \boldsymbol{D}_\sigma^{-1} \boldsymbol{C}_\sigma\right\}$ (dgees)

end

A. Parallelization over QR factorizations

In Section III, we have shown that the most relevant part of the VF scheme in terms of floating points operations is constituted by the QR factorizations of the matrices

$$\begin{bmatrix} \boldsymbol{\varphi}_1 & -\boldsymbol{S}_l\boldsymbol{\varphi}_1 \end{bmatrix} \tag{28}$$

with $l = i + (P-1)j$ spanning the considered set of responses $\forall (i,j) \in \mathcal{I}_\mu$ that will share the same poles in the rational model. Fortunately, each QR factorization can be computed independently once the matrix (28) is available. This leads to an obvious parallelization strategy, which introduces a parallel section that assigns individual QR factorizations to individual threads, according to some scheduling rule.

Algorithm 2 VF (QR parallelization, "none" splitting strategy)

Input

T : number of available threads
P : number of ports
N : number of poles
K : number of frequency samples
ω_k : frequency points, $\forall k \in \{1, \ldots, K\}$
$S_{ij,k}$: frequency samples, $\forall i, j \in \{1, \ldots, P\}$, $\forall k \in \{1, \ldots, K\}$

$z_n \leftarrow$ starting poles
repeat
  $p_n \leftarrow z_n$
  $z_n \leftarrow \text{POLESREF}(T, P^2, N, K, \mathcal{I}, p_n, \omega_k, S_{ij,k})$
until poles convergence $(|p_n - z_n| < \varepsilon)$
$\tilde{p}_n \leftarrow p_n$
Find $\tilde{r}_{ij,n}$ by solving (5) (`dgels`)

A pseudo-code corresponding to the parallelized pole relocation, also depicted in the inner loop of Figure 2, is listed in Algorithm 1. The function POLESREF accepts the set of initial poles p_n and the raw data samples, and provides on output a new set of relocated poles z_n using T computing threads. Since our implementation is based on the LAPACK libraries [53], we have also highlighted the relevant LAPACK functions used to solve the various algebraic problems involved in the algorithm.

The parallel section is introduced in the code by using the OpenMP directive

```
#pragma omp parallel for
```

that informs the compiler, in particular the preprocessor, that the following for-loop has to be executed in parallel. Following this directive, the OpenMP environment produces the required code for the creation and the synchronization of the threads. This directive allows also to specify the scheduling strategy to be adopted for the assignment of the various iteration steps to individual threads. Since in this case all matrices (28) have the same size, and consequently there are no load balancing issues since all QR factorizations will require the same CPU time, a static scheduling is sufficient to achieve the best performance. The result is as specified in Algorithm 1, with the t-th thread performing the QR factorization of (28) for $l = \left\lfloor \frac{t}{T} L \right\rfloor + 1, \left\lfloor \frac{t}{T} L \right\rfloor + 2, \ldots, \left\lfloor \frac{t+1}{T} L \right\rfloor$.

The parallelized code requires more memory than the corresponding serial code, since some variables and matrices can not be shared among the threads and need to be replicated. This is also highlighted in Algorithm 1, where the superscript $^{(t)}$ indicates that the corresponding memory location has to be private for each thread. The amount of memory required by the parallel QR section of the code is about $32TK(N+1)$ bytes.

This type of parallelization is most effective when the number of responses in each subsystem $\mathcal{I}_\mu$ is large with respect to the number of threads, i.e., $|\mathcal{I}_\mu| \gg T$. Therefore, the best performance of this parallelization is achieved with the "none" splitting strategy, i.e., when all P^2 responses are considered as a whole and will share the same pole set in the rational model. Even in such case, when P^2 is not an integral multiple of T, a small percentage of parallel efficiency (less than 1% when $P > 40$ and $M \leqslant 16$) is lost due to a slightly unbalanced workload arising from a different number of QR factorizations assigned to different threads (this difference is at most one). This loss is considered to be negligible. A pseudocode for the QR-parallelized VF scheme, using the "none" splitting strategy, is summarized in Algorithm 2.

B. Parallelization over responses

When a splitting scheme different from "none" is used, VF can be applied independently to each subsystem. In this case, a trivial and efficient way to parallelize the overall model extraction is to assign these individual VF runs to individual threads. In case the number of responses in each subset $\mathcal{I}_\mu$ is the same, a static scheduling can be applied, as for the QR parallelization strategy. This leads to the t-th thread executing a private VF run on the subsets $\mathcal{I}_\mu$, for $\mu = \left\lfloor \frac{t}{T} M \right\rfloor + 1, \left\lfloor \frac{t}{T} M \right\rfloor + 2, \ldots, \left\lfloor \frac{t+1}{T} M \right\rfloor$.

The best performance of this method is achieved when the number of partitions is largest, i.e. for the "all" partitioning scheme. In this case, the parallel efficiency is practically ideal, since all phases of VF run concurrently. Therefore, in comparison with the parallelization over QR factorizations, the speed-up factors are expected to be superior. This will be confirmed by the numerical results in Section V. A pseudocode of the parallelized VF over responses in the "all" splitting case is provided by Algorithm 3. Note that the POLESREF function is called on a single response using a single thread, since the parallelization is performed at a higher hierarchical level.

C. Mixed parallelization

Both QR-based and response-based parallelization schemes can be applied in case the adopted splitting strategy is such that $1 < |\mathcal{J}| < P^2$. The "column" partitioning scheme described in Section II-D is one such example, corresponding to a number P of independent subsystems, each requiring P independent QR factorizations. When the number of ports is much larger than the number of available threads $P \gg T$, good performance can be obtained by directly applying the parallel schemes described in Sections IV-A and IV-B. In the QR-based parallelization, each subsystem is processed sequentially, with all QR decompositions being computed by a multi-threaded run of the poles refinement stage. The corresponding pseudo-code is detailed in Algorithm 4. For the response-based parallelization, instead, blocks of multiple T different subsystems are analyzed concurrently by independent threads, which perform the required QR decompositions of the poles refinement step sequentially. The corresponding pseudo-code is detailed in Algorithm 5. The level of performance and parallel efficiency of these two alternative schemes is expected to be intermediate between the QR-based parallelization in the "none" splitting case and the response-based parallelization in the "all" splitting case.

A more advanced approach is conceivable for massively parallel model extraction of large-scale systems with several

Algorithm 3 VF (split parallelization, "all" splitting strategy)

Input

T : number of available threads
P : number of ports
N : number of poles
K : number of frequency samples
ω_k : frequency points, $\forall k \in \{1, \dots, K\}$
$S_{ij,k}$: frequency samples, $\forall i, j \in \{1, \dots, P\}$, $\forall k \in \{1, \dots, K\}$

parallel region

$t \leftarrow$ thread number
for $\mu \leftarrow \left\lfloor \frac{t}{T} P^2 \right\rfloor + 1$ **to** $\left\lfloor \frac{t+1}{T} P^2 \right\rfloor$
$\mathcal{I}_\mu \leftarrow \{(i,j)\}$, $\mu = i + (P-1)j$
$z_n^{(t)} \leftarrow$ starting poles
repeat
$p_n^{(t)} \leftarrow z_n^{(t)}$
$z_n^{(t)} \leftarrow \text{POLESREF}(1, 1, N, K, \mathcal{I}_\mu, p_n^{(t)}, \omega_k, S_{ij,k})$
until poles convergence $(|p_n^{(t)} - z_n^{(t)}| < \varepsilon)$
Find $\tilde{r}_{ij,n}$ by solving (5) (with $(i,j) \in \mathcal{I}_\mu$) (`dgels`)
end for

end parallel region

Algorithm 4 VF (QR parallelization, "column" splitting strategy)

Input

T : number of available threads
P : number of ports
N : number of poles
K : number of frequency samples
ω_k : frequency points, $\forall k \in \{1, \dots, K\}$
$S_{ij,k}$: frequency samples, $\forall i, j \in \{1, \dots, P\}$, $\forall k \in \{1, \dots, K\}$

for $\mu \leftarrow 1$ **to** P
$\mathcal{I}_\mu \leftarrow \{(1,\mu), \dots, (P,\mu)\}$
$z_n \leftarrow$ starting poles
repeat
$p_n \leftarrow z_n$
$z_n \leftarrow \text{POLESREF}(T, P, N, K, \mathcal{I}_\mu, p_n, \omega_k, S_{ij,k})$
until poles convergence $(|p_n - z_n| < \varepsilon)$
$\tilde{p}_n \leftarrow p_n$
Find $\tilde{r}_{ij,n}$ by solving (5) (with $(i,j) \in \mathcal{I}_\mu$) (`dgels`)
end for

hundred ports. Since the response-based and the QR-based parallelizations operate on different hierarchical levels, it is possible to nest the two parallelizations in a hybrid implementation, where the inner level will perform QR factorizations and the outer level will parallelize over subsystems. This mixed strategy is suitable for a hierarchical hardware architecture, e.g. a cluster of multi-core machines. Different

Algorithm 5 VF (split parallelization, "column" splitting strategy)

Input

T : number of available threads
P : number of ports
N : number of poles
K : number of frequency samples
ω_k : frequency points, $\forall k \in \{1, \dots, K\}$
$S_{ij,k}$: frequency samples, $\forall i, j \in \{1, \dots, P\}$, $\forall k \in \{1, \dots, K\}$

parallel region

$t \leftarrow$ thread number
for $\mu \leftarrow \left\lfloor \frac{t}{T} P \right\rfloor + 1$ **to** $\left\lfloor \frac{t+1}{T} P \right\rfloor$
$\mathcal{I}_\mu \leftarrow \{(1,\mu), \dots, (P,\mu)\}$
$z_n^{(t)} \leftarrow$ starting poles
repeat
$p_n^{(t)} \leftarrow z_n^{(t)}$
$z_n^{(t)} \leftarrow \text{POLESREF}(1, P, N, K, \mathcal{I}_\mu, p_n^{(t)}, \omega_k, S_{ij,k})$
until poles convergence $(|p_n^{(t)} - z_n^{(t)}| < \varepsilon)$
Find $\tilde{r}_{ij,n}$ by solving (5) (with $(i,j) \in \mathcal{I}_\mu$) (`dgels`)
end for

end parallel region

subsystems may be assigned to different machines, whereas individual machines will perform parallel QR decompositions on individual subsystems. A hybrid MPI-OpenMP software architecture is desirable for such an implementation. The results of this hybrid scheme will be documented in a forthcoming report.

V. Numerical results

We report in this section a set of numerical results in terms of speed-up and parallel efficiency of the various PVF formulations presented in Section IV. These results are discussed with reference to the benchmarks of Table I in different subsections, according to the adopted splitting strategy. In order to draw meaningful conclusions on a fair comparison between different implementations, a fixed number $H = 3$ of iterations is used in the poles relocation loop. In all cases, this number of iterations combined with the fixed order N (see the 4-th column of Table I) guarantees the final RMS error between extracted model and original data to be less than 10^{-3}. As a further evidence, Figure 4 depicts the model vs data responses for case 1, showing excellent correlation and no visible difference. The same level of accuracy was obtained for all cases.

All tests were performed on a IBM BladeCenter LS42 server with four quad-core AMD Opteron processors (2.3 GHz clock), summing to 16 cores, and 32 GB of RAM. A full scalability test was performed by running the algorithms with a variable number of threads $T = 1, \dots, 16$. Moreover, in order to obtain meaningful statistical results, each run was repeated 10 times, and the mean CPU time $\bar{\tau}^{(T)}$ was recorded.

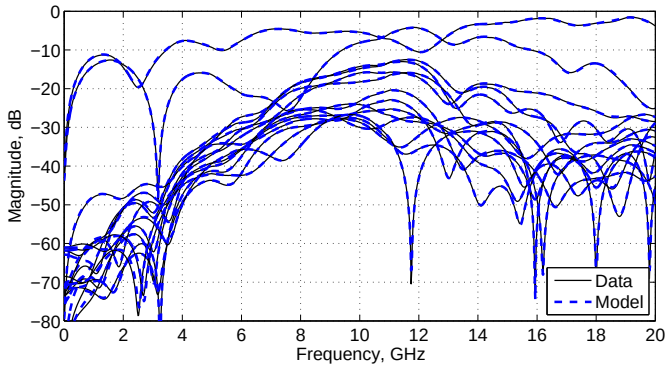

Fig. 4. Comparison between some model responses and the corresponding raw data for case 1.

A. "None" splitting strategy

Table III summarizes the results for the "none" splitting strategy. We recall that, in this case, only the parallelization over QR decompositions can be performed, since all responses are dealt with concurrently and share a common pole set. The table includes two sections. Columns 2–4 report the average CPU time (1 and 16 threads) required by the parallel section of the poles relocation phase, and the corresponding speedup factor. This section highlights that for the large-scale cases a nearly ideal speedup is achieved in the part of the code that is actually parallel. Some deviation from the ideal speedup is observed only for those cases that require less than 2–3 seconds for the single-thread run. The multi-threaded run for these cases requires fraction of a second, so that the speedup factor is masked by additional overheads related to memory management and operating system functioning.

Columns 5–7 of Table III report the overall (average) runtime and corresponding speedup factors for the complete model extraction, including also the VF modules that cannot be parallelized efficiently. Some additional small overhead is observed, but the overall performance of the PVF code is quite satisfactory, since the parallel efficiency for the most challenging cases (1–5) is in the range 70%–80%. All other cases, although characterized by a medium to large port count, require either a small order N or have few frequency samples K, so that the overall workload is smaller. For those cases, the important result is the total runtime of the parallel code (column 6), which is significantly less than one second in most cases.

Figure 5 shows a detailed scalability study for the largest examples, by reporting the speedup achieved by the parallel section and by the overall PVF run for a varying number of threads. These results confirm the nearly ideal speedup of the parallel section and the quite acceptable overall efficiency.

B. "All" splitting strategy

We report in Table IV the results of the parallelization in the "all" splitting case, i.e., when the P^2 responses are modeled independently by separate VF runs, which execute in blocks of T parallel instances. The table reports the average runtime achieved using 1 and 16 threads, respectively, and

TABLE III
CPU TIME (SECONDS) AND SPEED-UP, "NONE" SPLITTING STRATEGY APPLIED TO EACH BENCHMARK CASE OF TABLE I.

Id	$\bar{\tau}_{\mathrm{QR}}^{(1)}$	$\bar{\tau}_{\mathrm{QR}}^{(16)}$	Speedup	$\bar{\tau}^{(1)}$	$\bar{\tau}^{(16)}$	Speedup
1	127.79	8.42	15.17	133.78	10.79	12.40
2	160.66	10.15	15.82	167.59	12.89	13.00
3	106.84	6.72	15.90	122.16	11.31	10.80
4	39.48	2.53	15.62	41.90	3.47	12.08
5	23.97	1.53	15.64	25.32	1.96	12.93
6	6.26	0.41	15.35	7.68	0.77	10.00
7	6.58	0.43	15.12	7.32	0.64	11.47
8	2.22	0.16	14.14	2.62	0.27	9.70
9	4.76	0.32	14.98	5.30	0.48	10.99
10	4.12	0.28	14.62	5.15	0.56	9.18
11	3.88	0.28	13.72	4.33	0.62	7.01
12	4.16	0.30	14.10	5.09	0.83	6.15
13	2.78	0.22	12.91	3.09	0.45	6.93
14	2.98	0.23	13.15	3.27	0.45	7.18
15	1.58	0.13	11.87	1.66	0.19	8.81
16	1.00	0.10	10.41	1.07	0.15	7.16
17	0.86	0.09	9.62	0.90	0.12	7.41
18	0.60	0.07	8.72	0.75	0.15	5.09
19	0.56	0.07	8.20	0.68	0.18	3.75
20	0.47	0.06	7.56	0.57	0.17	3.46
21	0.30	0.05	5.68	0.34	0.09	3.94
22	0.70	0.07	9.80	0.86	0.15	5.52
23	0.13	0.03	4.07	0.15	0.05	2.93
24	0.07	0.03	2.09	0.08	0.04	1.82

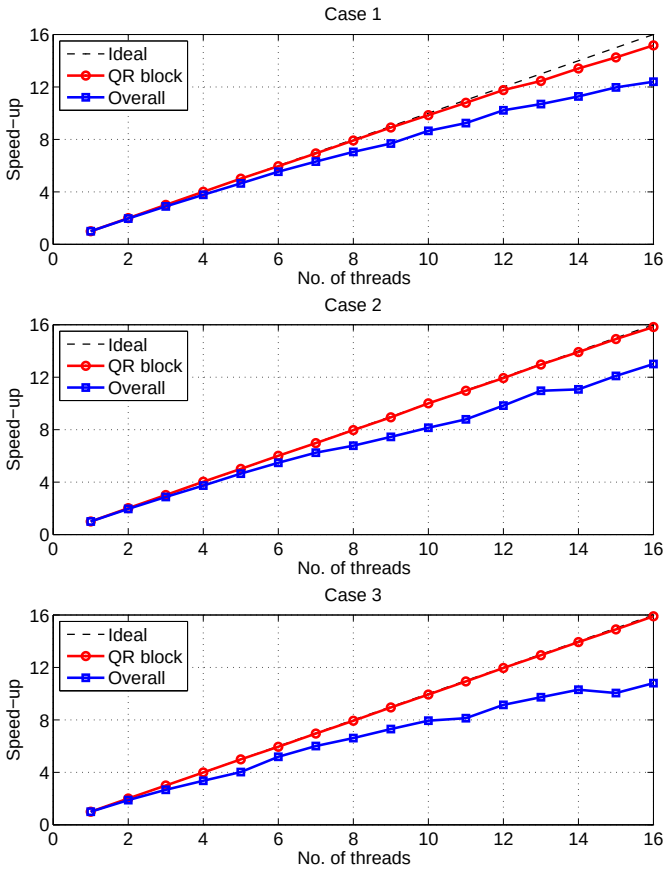

Fig. 5. Speed-up diagrams for the first 3 cases of Table I ("none" splitting strategy).

TABLE IV
CPU TIME (SECONDS) AND SPEED-UP, "ALL" SPLITTING STRATEGY APPLIED TO EACH BENCHMARK CASE OF TABLE I.

Id	$\bar{\tau}^{(1)}$	$\bar{\tau}^{(16)}$	Speedup
1	155.46	10.26	15.15
2	216.21	14.23	15.19
3	194.30	12.88	15.09
4	57.15	3.84	14.89
5	34.66	2.31	15.02
6	15.64	1.04	15.07
7	9.97	0.67	14.82
8	5.43	0.36	15.08
9	7.19	0.48	14.83
10	14.46	1.00	14.48
11	6.04	0.41	14.74
12	10.47	0.76	13.87
13	4.68	0.42	11.22
14	4.47	0.38	11.67
15	2.97	0.25	11.70
16	1.87	0.18	10.54
17	1.61	0.14	11.28
18	1.16	0.13	8.92
19	1.04	0.12	8.70
20	0.88	0.11	8.30
21	0.90	0.09	10.17
22	3.76	0.46	8.10
23	0.72	0.11	6.49
24	0.38	0.08	4.53

the corresponding speedup factor. Figure 6 depicts a complete scalability study on a varying number of threads for the largest cases. These results collectively show that

- the total runtime is slightly larger in the "all" with respect to the "none" splitting case, thus confirming the theoretical analysis of Section III;
- the overall speedup factors are better than for the "none" splitting case. This is also expected, since the fraction of code that is parallelized in the "all" case is larger.

Despite the embarrassingly parallel implementation, some small overhead remains also for the largest cases. This can be explained noting that system calls for memory management need in some cases exclusive access. This is achieved by imposing implicit barriers that may block the execution of some threads until memory access is released. In any case, this overhead can be considered to be negligible given the very small overall runtime.

C. "Column" splitting strategies

In the "column" splitting case, both QR- and response-based parallelization schemes can be applied. The results of these two approaches are presented in Tables V and VI, respectively, and compared for the largest cases in Fig. 7. The QR-based parallelization is the least efficient, both in terms of speedup and total runtime, with respect to the "none" and the "all" splitting cases. This is readily explained by noting that only P parallel instances are available (compared to P^2 of the other schemes). The subdivision of these P instances into T threads is therefore more likely to be unbalanced. This fact is demonstrated by the typical staircase behavior of the corresponding speedup factors in Fig. 7. This behavior arises since some threads will execute $\lfloor \frac{P}{T} \rfloor$ runs, and some other threads will run $\lfloor \frac{P}{T} \rfloor + 1$ runs whenever P is not an integral multiple of T. The result is an unbalanced load which depends on the number of concurrent threads T.

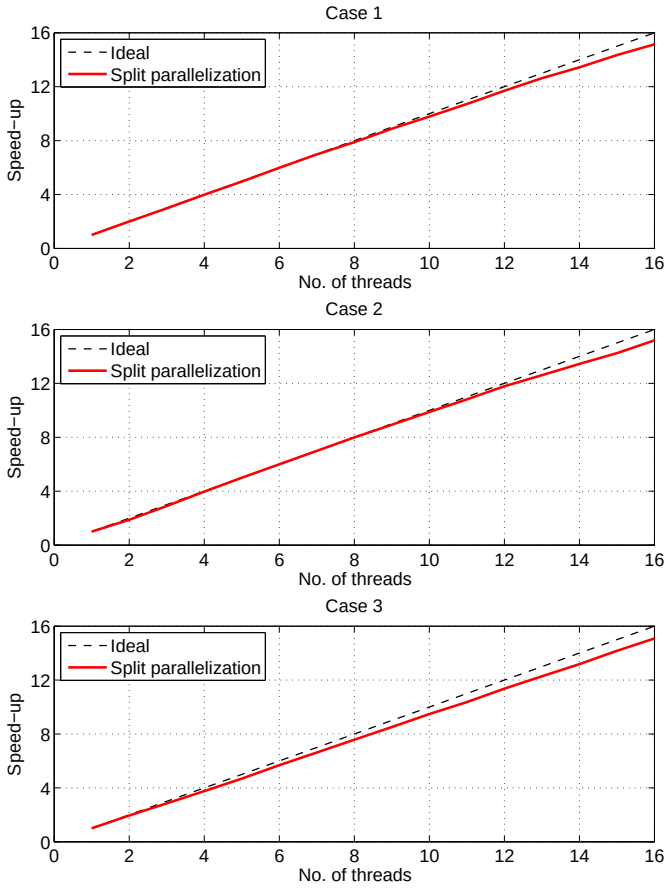

Fig. 6. Speed-up diagrams for the first 3 cases of Table I ("all" splitting strategy).

Let us consider now the performance of the response-based parallelization in the "column" splitting case. The parallel efficiency and the speedup factors are significantly improved with respect to the QR-based parallelization. This is reasonable, since the entire VF scheme is run concurrently for each subset of responses, in an embarrassingly parallel way. The same granularity issues affect the speedup factors, see Fig. 7, since only P instances are available. Therefore, as expected, the speedup factors are not as good as for the "none" and "all" splitting cases. Nevertheless, the total runtime (over 16 threads) is the best among all implementations, as demonstrated by Table VII. This is justified by noting that each transfer matrix column is assigned to a dedicated computing thread, which processes all individual blocks of the VF algorithm. Therefore, a much larger fraction of the code is parallelized, including not only the QR part (14), but also the LSP problems (15) and (5).

VI. CONCLUSIONS

This paper introduced a number of different Parallel Vector Fitting (PVF) schemes for the extraction of rational macro-models from tabulated frequency responses of linear time-invariant systems. The various schemes are different for what

TABLE V
CPU TIME (SECONDS) AND SPEED-UP, "COLUMN" SPLITTING STRATEGY WITH QR PARALLELIZATION APPLIED TO EACH BENCHMARK CASE OF TABLE I.

Id	$\bar{\tau}_{QR}^{(1)}$	$\bar{\tau}_{QR}^{(16)}$	Speedup	$\bar{\tau}^{(1)}$	$\bar{\tau}^{(16)}$	Speedup
1	127.97	11.84	10.81	133.07	14.08	9.45
2	159.39	13.96	11.42	163.84	17.01	9.63
3	106.05	7.98	13.28	109.83	10.92	10.06
4	39.71	4.24	9.38	41.90	6.34	6.61
5	23.88	2.34	10.23	24.46	3.81	6.42
6	6.16	0.57	10.88	6.50	1.23	5.29
7	6.62	0.82	8.08	6.94	1.59	4.36
8	2.21	0.25	8.99	2.38	0.64	3.73
9	4.76	0.64	7.48	4.99	1.20	4.14
10	4.17	0.32	13.23	4.41	0.83	5.34
11	3.89	0.44	8.79	4.07	0.79	5.18
12	4.11	0.32	12.98	4.41	0.67	6.55
13	2.78	0.41	6.78	2.91	0.81	3.58
14	2.95	0.46	6.37	3.09	1.26	2.45
15	1.57	0.25	6.30	1.65	0.45	3.70
16	1.01	0.24	4.25	1.07	0.47	2.25
17	0.86	0.13	6.86	0.91	0.21	4.40
18	0.60	0.08	7.13	0.66	0.19	3.54
19	0.55	0.09	5.95	0.61	0.18	3.46
20	0.47	0.08	6.03	0.52	0.17	3.08
21	0.30	0.05	6.28	0.33	0.11	3.00
22	0.73	0.10	7.45	0.80	0.22	3.65
23	0.13	0.32	0.40	0.15	0.39	0.37
24	0.07	0.23	0.31	0.08	0.29	0.29

TABLE VI
CPU TIME (SECONDS) AND SPEED-UP, "COLUMN" SPLITTING STRATEGY, RESPONSE-BASED PARALLELIZATION APPLIED TO EACH BENCHMARK CASE OF TABLE I.

Id	$\bar{\tau}^{(1)}$	$\bar{\tau}^{(16)}$	Speedup
1	134.45	10.16	13.23
2	164.53	12.23	13.45
3	112.29	7.30	15.39
4	42.21	3.74	11.28
5	24.63	1.60	15.40
6	6.90	0.46	14.90
7	6.95	0.55	12.64
8	2.53	0.20	12.70
9	5.00	0.46	10.87
10	4.54	0.31	14.85
11	4.07	0.33	12.17
12	4.46	0.32	13.80
13	2.93	0.22	13.13
14	3.11	0.28	11.04
15	1.70	0.16	10.77
16	1.09	0.11	9.58
17	0.93	0.10	9.49
18	0.66	0.08	8.44
19	0.61	0.07	9.10
20	0.52	0.06	8.94
21	0.35	0.04	9.38
22	0.84	0.06	13.80
23	0.15	0.04	4.33
24	0.09	0.01	5.85

concerns the choice of parallelization strategy. Yet, their performance is comparable with minor differences. We can safely conclude that the parallel efficiency and speedup factors for the code fragments that have been parallelized are close to ideal for medium- and large-scale structures. As expected, the speedup generally degrades when the problem size decreases. For small-scale structures, a parallel implementation is not needed at all, since total runtime of standard (serial) VF is negligible. The proposed parallel schemes thus result most ef-

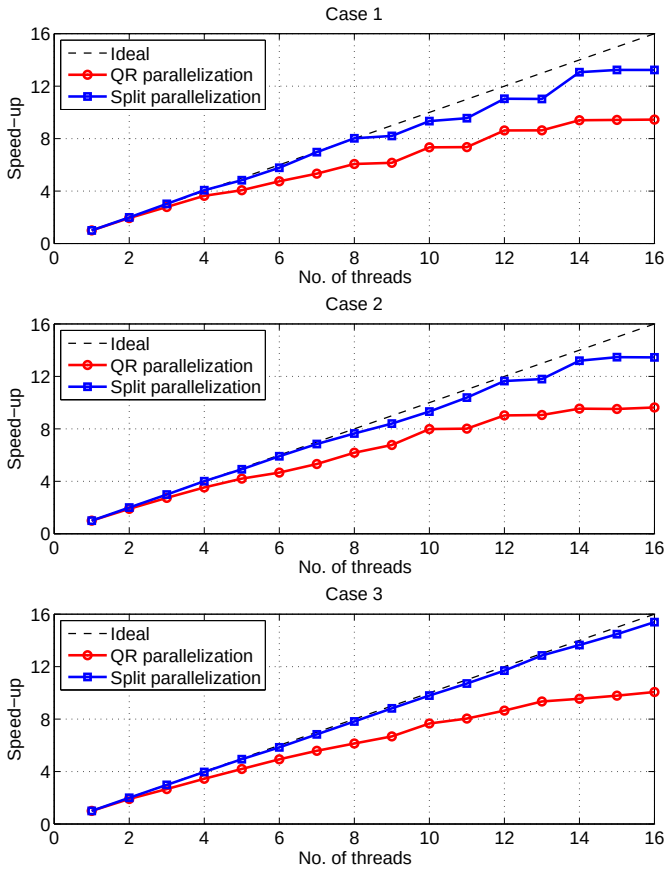

Fig. 7. Speed-up diagrams for the first 3 cases of Table I ("column" splitting strategy).

TABLE VII
CPU TIME (SECONDS) OF ALL PARALLEL IMPLEMENTATIONS OF THE VF ALGORITHM (16 THREADS) FOR THE BENCHMARK CASES OF TABLE I.

Id	None	Col-QR	Col-Resp.	All
1	10.79	14.08	10.16	10.26
2	12.89	17.01	12.23	14.23
3	11.31	10.92	7.30	12.88
4	3.47	6.34	3.74	3.84
5	1.96	3.81	1.60	2.31
6	0.77	1.23	0.46	1.04
7	0.64	1.59	0.55	0.67
8	0.27	0.64	0.20	0.36
9	0.48	1.20	0.46	0.48
10	0.56	0.83	0.31	1.00
11	0.62	0.79	0.33	0.41
12	0.83	0.67	0.32	0.76
13	0.45	0.81	0.22	0.42
14	0.45	1.26	0.28	0.38
15	0.19	0.45	0.16	0.25
16	0.15	0.47	0.11	0.18
17	0.12	0.21	0.10	0.14
18	0.15	0.19	0.08	0.13
19	0.18	0.18	0.07	0.12
20	0.17	0.17	0.06	0.11
21	0.09	0.11	0.04	0.09
22	0.15	0.22	0.06	0.46
23	0.05	0.39	0.04	0.11
24	0.04	0.29	0.01	0.08

ficient for large-size models, for which nearly ideal efficiency and scalability is achieved.

Despite some small overheads that are intrinsic in the basic VF algorithm structure, the overall runtime is drastically reduced. As a result, very large scale structures can be processed by the proposed PVF schemes, leading to model extractions in few seconds even for the most challenging cases. Future work will address hybrid MPI-OpenMP software architectures, for massively parallel deployment on clusters of multi-core machines. Another direction with enormous potential will be porting the parallel VF code to GPU-based architectures, to investigate on scalability of VF schemes on much larger number of cores/threads than currently available on commodity hardware.

VII. Acknowledgments

The Authors are grateful to Dr. Katopis and Dr. Becker (IBM) for supporting this activity through an IBM Shared University Research (SUR) Grant. This work was supported in part by the Italian Ministry of University (MIUR) under a Program for the Development of Research of National Interest (PRIN grant #2008W5P2K) and in part by IdemWorks s.r.l.

References

[1] W. H. Schilders, H. A. van der Vorst, J. Rommes (Eds.), *Model Order Reduction: Theory, Research Aspects and Applications*, Springer-Verlag, 2008.

[2] A. C. Antoulas, *Approximation of large-scale dynamical systems*, SIAM, Philadelphia, 2005.

[3] B. C. Moore, "Principal component analysis in linear systems," *IEEE Trans. Automat. Contr.*, vol. AC-26, pp. 17-32, Feb. 1981.

[4] P. Feldmann and R. W. Freund, "Efficient linear circuit analysis by Pade approximation via the Lanczos process," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 5, pp. 639-649, May 1995.

[5] L. M. Silveira, M. Kamon, I. Elfadel and J. White, "A coordinate transformed Arnoldi algorithm for generating stable reduced-order models of arbitrary RLC circuits," in *IEEE/ACM Proc. ICCAD*, Nov. 1996, pp. 288-294.

[6] A. Odabasioglu, M. Celik and L. T. Pileggi, "PRIMA: Passive reduced-order interconnect macromodeling algorithm," *IEEE Trans. Computer-Aided Design*, vol. 17, no. 8, pp. 645–654, Aug. 1998.

[7] G. Shi and C. J. R. Shi, "Model order reduction by dominant subspace projection: error bound, subspace computation and circuit application," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 5, pp. 975–993, May 2005.

[8] J. R. Phillips and L. M. Silveira, "Poor Mans TBR: a simple model reduction scheme," *IEEE Trans. Comput.-Aided Design*, vol. 24, no. 1, pp. 43–55, Jan 2005.

[9] D. Li, S. X.-D. Tan, and B. McGaughy, "ETBR: Extended truncated balanced realization method for on-chip power grid network analysis," in *Proc. European Design and Test Conf.*, 2008, pp. 432–437.

[10] P. Feldman, "Model order reduction techniques for linear systems with large number of terminals," in *Proc. European Design and Test Conf.*, 2004, pp. 944–947.

[11] P. Li and W. Shi, "Model order reduction of linear networks with massive ports via frequency-dependent port packing," in *Proc. Design Autom. Conf.*, 2006, pp. 267–272.

[12] B. Yan, L. Zhou, S. X.-D. Tan, J. Chen, and B. McGaughy, "DeMOR: decentralized model order reduction of linear networks with passive ports," in *Proc. Design Autom. Conf.*, 2008, pp. 409–414.

[13] I. M. Elfadel and D. L. Ling, "A block rational Arnoldi algorithm for multipoint passive model order reduction of multiport RLC networks," in *Proc. Int. Conf. Computer Aided Design*, Nov. 1997, pp. 66–71.

[14] B. Gustavsen, A. Semlyen, "Rational approximation of frequency responses by vector fitting," *IEEE Trans. Power Delivery*, Vol. 14, N. 3, July 1999, pp. 1052–1061.

[15] B. Gustavsen, "Computer code for rational approximation of frequency dependent admittance matrices," *IEEE Trans. Power Delivery*, Vol. 17, N. 4, October 2002, pp. 1093–1098.

[16] B. Gustavsen, A. Semlyen, "A robust approach for system identification in the frequency domain," *IEEE Trans. Power Delivery*, Vol. 19, N. 3, July 2004, pp. 1167–1173.

[17] B. Gustavsen, "Improving the pole relocating properties of vector fitting," *IEEE Trans. Power Delivery*, Vol. 21, N. 3, Aug. 2006, pp. 1587–1592.

[18] D. Deschrijver, B. Haegeman, T. Dhaene, "Orthonormal Vector Fitting: A Robust Macromodeling Tool for Rational Approximation of Frequency Domain Responses," *IEEE Trans. Adv. Packaging*, vol. 30, pp. 216–225, May 2007.

[19] S. Grivet-Talocia, M. Bandinu, "Improving the Convergence of Vector Fitting in Presence of Noise", *IEEE Transactions on Electromagnetic Compatibility*, vol. 48, n. 1, pp. 104-120, February, 2006.

[20] F. Ferranti, Y. Rolain, L. Knockaert, and T. Dhaene, "Variance Weighted Vector Fitting for Noisy Frequency Responses," in IEEE Microwave and Wireless Components Letters, vol. 20, no. 4, pp. 187–189, April 2010.

[21] S. Grivet-Talocia, "Package macromodeling via Time-Domain Vector Fitting," *IEEE Microwave Wireless Comp. Lett.*, Vol. 13, No. 11, 2003.

[22] Y. S. Mekonnen, J. Schutt-Ainé, "Broadband macromodeling of sampled frequency data using z-domain vector-fitting method," em IEEE Workshop on Signal Propagation on Interconnects, 13-16 May 2007, pp. 45–48.

[23] A. Chinea, P. Triverio, S. Grivet-Talocia, "Delay-Based Macromodeling of Long Interconnects from Frequency-Domain Terminal Responses," *IEEE Transactions on Advanced Packaging*, Vol. 33, No. 1, pp. 246–256, Feb. 2010.

[24] D. Deschrijver, M. Mrozowski, T. Dhaene, D. De Zutter, "Macromodeling of Multiport Systems Using a Fast Implementation of the Vector Fitting Method," *IEEE Microwave and Wireless Components Letters*, Vol. 18, N. 6, June 2008, pp.383–385.

[25] S. Lefteriu, A. C. Antoulas, "A New Approach to Modeling Multiport Systems From Frequency-Domain Data," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 29, no. 1, pp. 14–27, Jan. 2010

[26] A. Chinea and S. Grivet-Talocia, "A parallel vector fitting implementation for fast macromodeling of highly complex interconnects," in *IEEE* 19^{th} *Topical Meeting on Electrical Performance of Electronic Packaging and Systems (EPEPS 2010), Austin, TX*, October 24–27, 2010.

[27] M. R. Wohlers, *Lumped and Distributed Passive Networks*, Academic Press, 1969.

[28] B. D. O. Anderson and S. Vongpanitlerd, *Network Analysis and Synthesis*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[29] P. Triverio, S. Grivet-Talocia, M. S. Nakhla, F. Canavero, R. Achar, "Stability, Causality, and Passivity in Electrical Interconnect Models," *IEEE Transactions on Advanced Packaging*, Vol. 30, No. 4, pp. 795–808, Nov. 2007.

[30] S. Grivet-Talocia, "Passivity enforcement via perturbation of Hamiltonian matrices", *IEEE Trans. CAS-I*, pp. 1755-1769, vol. 51, n. 9, September, 2004

[31] S. Grivet-Talocia, A. Ubolli "On the Generation of Large Passive Macromodels for Complex Interconnect Structures", *IEEE Trans. Adv. Packaging*, vol. 29, No. 1, pp. 39–54, Feb. 2006

[32] D. Saraswat, R. Achar and M. Nakhla, "Global Passivity Enforcement Algorithm for Macromodels of Interconnect Subnetworks Characterized by Tabulated Data", *IEEE Transactions on VLSI Systems*, Vol. 13, No. 7, pp. 819–832, July 2005.

[33] C. P. Coelho, J. Phillips, L. M. Silveira, "A Convex Programming Approach for Generating Guaranteed Passive Approximations to Tabulated Frequency-Data", *IEEE Trans. Computed-Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 2, February 2004, pp. 293–301.

[34] H. Chen, J. Fang, "Enforcing Bounded Realness of S parameter through trace parameterization", in *12th IEEE Topical Meeting on Electrical Performance of Electronic Packaging*, October 27–29, 2003, Princeton, NJ, pp. 291–294.

[35] B. Gustavsen, A. Semlyen, "Enforcing passivity for admittance matrices approximated by rational functions", *IEEE Trans. Power Systems*, Vol. 16, N. 1, Feb. 2001, pp. 97–104.

[36] B. Gustavsen, "Computer Code for Passivity Enforcement of Rational Macromodels by Residue Perturbation," *IEEE Trans. Adv. Packaging*, vol. 30, pp. 209–215, May 2007.

[37] D. Saraswat, R. Achar and M. Nakhla, "A Fast Algorithm and Practical Considerations For Passive Macromodeling Of Measured/Simulated Data", *IEEE Transactions on Components, Packaging and Manufacturing Technology*, Vol. 27, N. 1, pp. 57–70, Feb. 2004.

[38] S. Grivet-Talocia, "An adaptive sampling technique for passivity characterization and enforcement of large interconnect macromodels," *IEEE Trans. Adv. Packaging*, vol. 30, pp. 226–237, May 2007.

[39] A Lamecki and M. Mrozowski, "Equivalent SPICE Circuits With Guaranteed Passivity From Nonpassive Models," IEEE Transactions on Microwave Theory And Techniques, Vol. 55, No. 3, March 2007, pp. 526–532.
[40] S. Grivet-Talocia, A. Ubolli "Passivity Enforcement with Relative Error Control", *IEEE Trans. on Microwave Theory and Techniques*, vol. 55, No. 11, November 2007, pp. 2374–2383.
[41] Z. Ye, L. M. Silveira, and J. R. Phillips, "Fast and Reliable Passivity Assessment and Enforcement with Extended Hamiltonian Pencil," in *International Conference on Computer Aided Design*, 2009, pp. 774–778.
[42] Z. Ye, L. M. Silveira, and J. R. Phillips, "Extended Hamiltonian Pencil for Passivity Assessment and Enforcement for S-parameter Systems," in *DATE 2010 Conference*, pp. 1148–1152.
[43] Z. Zhang, C. U. Lei, and N. Wong, "GHM: A generalized Hamiltonian method for passivity test of impedance/admittance descriptor systems," in *Proc. Int. Conf. Comput.-Aided Design, San Jose, CA*, Nov. 2009, pp. 767–773.
[44] Z. Zhang and N. Wong, "Passivity test of immittance descriptor systems based on generalized Hamiltonian methods," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 57, no. 1, pp. 61–65, Jan. 2010.
[45] L. Gobbato, A. Chinea, S. Grivet-Talocia, "A Parallel Hamiltonian Eigensolver for Passivity Characterization and Enforcement of Large Interconnect Macromodels," DATE 2011 Conference, to appear.
[46] P. Stoica, T. Soderstrom,"The Steiglitz-McBride identification algorithm revisited–Convergence analysis and accuracy aspects," *IEEE Trans. Automatic Control*, vol. 26, no. 3, pp. 712- 717, Jun 1981.
[47] G. H. Golub, C. F. Van Loan, *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.
[48] R. Achar, M. Nakhla, "Minimum realization of reduced-order high-speed interconnect macromodels," in *Signal Propagation on Interconnects*, H. Grabinski and P. Nordholz Eds., Kluwer, 1998.
[49] F. Ebert, T. Stykel, "Rational interpolation, minimal realization and model reduction," Research Center MATHEON, Berlin, Germany, Tech. Rep. 371-2007, 2007.
[50] *HSPICE Applications Manual*, Available: www.synopsys.com
[51] OpenMP Architecture Review Board, *OpenMP C and C++ Application Program Interface - Version 2.0*, Mar. 2002. Available: www.openmp.org/mp-documents/cspec20.pdf.
[52] W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir, M. Snir, *MPI: The Complete Reference*, Sep. 1998, Available: www.netlib.org/utk/papers/mpi-book/mpi-book.html
[53] *LAPACK Users' Guide - Third edition*, Aug. 1999, Available: www.netlib.org/lapack/lug

Alessandro Chinea received the Laurea Specialistica (M.Sc.) and Ph.D. degrees in electronic engineering from Politecnico di Torino, Italy in 2006 and 2010, respectively. Since 2007, he joined the EMC group where he is currently a research assistant. In 2009 he spent a period at the Department of Information Technology (INTEC) of the Ghent University, Belgium, working under the supervision of the professors T. Dhaene and L. Knockaert. His research interests concern passive macromodeling of electrical interconnects for electromagnetic compatibility and signal/power integrity problems. Dr. Chinea received the Optime Award from the Unione Industriale di Torino and he was selected for the IBM EMEA Best Student Recognition Event 2006.

Stefano Grivet-Talocia (M'98–SM'07) received the Laurea and the Ph.D. degrees in electronic engineering from Politecnico di Torino, Italy. From 1994 to 1996, he was with the NASA/Goddard Space Flight Center, Greenbelt, MD, USA. Currently, he is an Associate Professor of Circuit Theory with Politecnico di Torino. His research interests are in passive macromodeling of lumped and distributed interconnect structures, modeling and simulation of fields, circuits, and their interaction, wavelets, time-frequency transforms, and their applications. He is author of more than 120 journal and conference papers. He is co-recipient of the 2007 Best Paper Award of the IEEE Trans. Advanced Packaging. He received the IBM Shared University Research (SUR) Award in 2007, 2008 and 2009. Dr. Grivet-Talocia served as Associate Editor for the IEEE TRANSACTIONS ON ELECTROMAGNETIC COMPATIBILITY from 1999 to 2001.